





Introdução à Linguagem Python

Bem-vindo ao nosso estudo sobre a Introdução à Linguagem de Programação Python. Este estudo ajudará você a compreender melhor como programar na linguagem Python, utilizando os tipos de dados string, lista, tupla e dicionário, que são muito importantes quando estamos trabalhando com análise de dados. Bons estudos!

Dados tipo String

A linguagem de programação Python, como outras linguagens, trabalha com tipos primitivos de dados, como inteiro, real e lógico e com cadeias de caracteres, chamadas de String. A String é um tipo de dado que comporta valores alfanuméricos e é muito utilizada nas operações de Inteligência Artificial e Análise de Dados.

As strings são sequências de caracteres, onde podemos acessar um caractere em uma dada posição utilizando um índice. No exemplo a seguir, caso se queira obter o caractere na primeira posição da string nome, basta acessar o índice 0 da variável. E podemos fatiar a string a partir de seus índices. Veja abaixo exemplos de fatiamento com a string nome.



```
File  Edit  Format  Run  Options  Window  Help
nome = 'Maria da Silva'
print(nome[0])    # M
print(nome[0:5])  # Maria
print(nome[1:])   # aria da Silva
print(nome[-1])   # a
print(nome[:-1])  # Maria da Silv
print(nome[-3])   # l
```

O que está com comentário é o resultado do respectivo comando. Veja que o primeiro print mostra o primeiro caractere da string nome, pois está no índice 0. No segundo print será mostrado o trecho da string que fica entre o índice 0 e o índice 4.

O comando print(nome[1:]) mostra a string a partir do índice 1, portanto o primeiro caractere não aparece. O comando print(nome[-1]) mostra o último caractere. O comando print(nome[:-1]) mostra todos os caracteres menos o último. O comando print(nome[-3]) mostra o terceiro caractere de trás para frente.

A string, por ser um módulo da linguagem, contém funções (chamadas de métodos) interessantes para trabalharmos com esse dado. Seguem alguns métodos importantes para manipular uma string.

Métodos de String

Método	Parâmetros	Descrição
upper	nenhum	Retorna uma string todo em maiúsculas
lower	nenhum	Retorna uma string todo em minúsculas

capitalize	nenhum	Retorna uma string com o primeiro caractere em maiúscula, e o resto em minúsculas
strip	nenhum	Retorna uma string removendo caracteres em branco do início e do fim
lstrip	nenhum	Retorna uma string removendo caracteres em brando do início
rstrip	nenhum	Retorna uma string removendo caracteres em brando do fim
count	item	Retorna o número de ocorrências de item
replace	old, new	Substitui todas as ocorrências da substring old por new
center	largura	Retorna uma string centrado em um campo de tamanho largura
ljust	largura	Retorna uma string justificado à esquerda em um campo de tamanho largura
rjust	largura	Retorna uma string justificado à direita em um campo de tamanho largura
find	item	Retorna o índice mais à esquerda onde a substring item é encontrada. Se não encontrar, retorna -1
rfind	item	Retorna o índice mais à direita onde a substring item é encontrada
index	item	Como find, mas causa um erro em tempo de execução caso item não seja encontrado
rindex	item	Como rfind, mas causa um erro em tempo de execução caso item não seja encontrado



Exemplos de utilização de alguns métodos:

File Edit Format Run Options Window Help

```
nome = 'Maria da Silva'
print(nome[0])    # M
print(nome[0:5]) # Maria
print(nome[1:])   # aria da Silva
print(nome[-1])   # a
print(nome[:-1]) # Maria da Silv
print(nome[-3])   # l
```

Para praticar, teste outros métodos e verifique os resultados. Esses métodos são muito utilizados na programação com dados alfanuméricos.

Em Python temos estruturas de dados muito utilizadas em Análise de Dados. Entre elas, as mais utilizadas são listas, tuplas e dicionários. Vamos mostrar a utilização dessas estruturas.

Listas

Listas são um tipo de variável que permite armazenar vários valores, que são acessados por um índice. Os valores de uma lista são mutáveis, isto é, podem ser alterados.

Criando uma lista:

L = [ ] # cria uma lista vazia

X = [10, 15, 20] # cria uma lista com 3 elementos



Operações com listas:

Adicionando um elemento da lista: para adicionar um elemento na lista usamos o comando append. Exemplo:

x.append(30) # inclui o número 30 no final da lista.

Removendo elementos da lista: para remover um elemento da lista usamos o comando del. Exemplo:

del l[0] # exclui o primeiro elemento da lista l.

Fatiando listas: O fatiamento de listas é feito da mesma forma que o fatiamento de string.

Exemplo:

L1 = [1,2,3,4,5]

print(L[1:4]) # 2,3,4

Tamanho da lista: para saber o tamanho da lista utilizamos o comando len.

Exemplo:

print(len(L)) # 5

Exemplo com algumas operações com lista:



```
File  Edit  Format  Run  Options  Window  Help
# Criando listas:

l = []  # cria uma lista vazia
x = [10, 15, 20]  # cria uma lista com três elementos
x.append(int(input('Digite um número para a lista: '))) # 8

# imprimindo cada elemento da lista:
print(x[0])  # 10
print(x[1])  # 15
print(x[2])  # 20
# incluindo um valor para a lista l:
l.append('José')
l.append('Maria')

print(x)  # [10, 15, 20, 8]
print(l)  # ['José', 'Maria']

print(x[1:3])  # [15,20]
print(len(x))  # 4
del(x[1])
print(x)  # [10, 20, 8] ,pois excluiu o 15, que estava no índice 1
```

Resultado do programa acima:

```
Digite um número para a lista: 8
10
15
20
[10, 15, 20, 8]
['José', 'Maria']
[15, 20]
4
[10, 20, 8]
```

Tuplas

Tuplas são parecidas com listas, com uma grande diferença. As tuplas são imutáveis. São ideais para representar listas de valores constantes. São criadas de forma sem... ...te às listas, mas são utilizados parênteses em vez de colchetes.

Exemplo:

Tupla = (1, 2, 3, 4, 5)

Podemos criar as tuplas sem utilizar parênteses:

T1 = 10, 20, 30, 40, 50

As tuplas podem ser particionadas como as listas, mas não podem ser incluídos valores após a sua criação. Porém se tiver uma lista dentro da tupla, a lista pode ser alterada.

Exemplo com tuplas:

File Edit Format Run Options Window Help

```
tupla = (1, 2, 3, 4, 5)
t1 = 'a', 'b', 'c', 40, 50
print(tupla)
print(t1)

# particionando a tupla:
print(tupla[1:4])
# tupla com lista:
t2 = (10, [20, 30, 40])
print('Tamanho da tupla: ',len(t2))
t2[1].append(50)
print(t2)
```

Resultado da execução do programa acima:



```
(1, 2, 3, 4, 5)
('a', 'b', 'c', 40, 50)
(2, 3, 4)
Tamanho da tupla:  2
(10, [20, 30, 40, 50])
```

Dicionários:

Dicionários têm uma estrutura similar às listas, mas o acesso a seus valores é diferente. Um dicionário é composto por um conjunto de chaves e valores. O dicionário, em Python, é criado utilizando chaves.

Exemplo:

produtos = {'Lápis': 1.50, 'Caderno': 15.00, 'Caneta': 2.50, 'Borracha': 5.00}

Para alterar um valor: produtos['Caderno'] = 20.00

Se encontrar a chave, o valor é alterado. Se não encontrar a chave, esta é incluída no dicionário. Atenção: as chaves não podem ser alteradas.

Para excluir: del(produtos['Caderno'])

Exemplo de programa com dicionário:

```
File Edit Format Run Options Window Help
produtos = {'Lápis': 1.50,
            'Caderno': 15.00,
            'Caneta': 2.50,
            'Borracha': 5.00}
print(produtos)
# alterando o preço do caderno:
produtos['Caderno'] = 20.00
print(produtos)
#incluindo um produto:
produtos['Livro'] = 50.00
print(produtos)
print('Preço do lápis:', produtos['Lápis'])
# verificando se uma chave existe:
print('Caderno' in produtos)
print('Gibi' in produtos)
print(produtos.keys())  # mostra as chaves
print(produtos.values()) # mostra os valores
print(produtos.items())
del (produtos['Caderno'])
print(produtos)
```

Resultado da execução do programa acima:

```
{'Lápis': 1.5, 'Caderno': 15.0, 'Caneta': 2.5, 'Borracha': 5.0}
{'Lápis': 1.5, 'Caderno': 20.0, 'Caneta': 2.5, 'Borracha': 5.0}
{'Lápis': 1.5, 'Caderno': 20.0, 'Caneta': 2.5, 'Borracha': 5.0, 'Livro': 50.0}
Preço do lápis: 1.5
True
False
dict_keys(['Lápis', 'Caderno', 'Caneta', 'Borracha', 'Livro'])
dict_values([1.5, 20.0, 2.5, 5.0, 50.0])
dict_items([('Lápis', 1.5), ('Caderno', 20.0), ('Caneta', 2.5), ('Borracha', 5.0
), ('Livro', 50.0)])
{'Lápis': 1.5, 'Caneta': 2.5, 'Borracha': 5.0, 'Livro': 50.0}
```

Atividade Extra

Para se aprofundar no assunto desta aula, leia sobre listas no livro Introdução à Programação com Python, de Nilo Ney Coutinho Menezes, capítulo 6.

Referência Bibliográfica



MENEZES, Nilo N. C. Introdução à Programação com Python: Algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2014.

SUMMERFIELD, Mark. Programação em Python 3: uma introdução completa à linguagem Python. Rio de Janeiro: Alta Books, 2012.

Ir para exercício